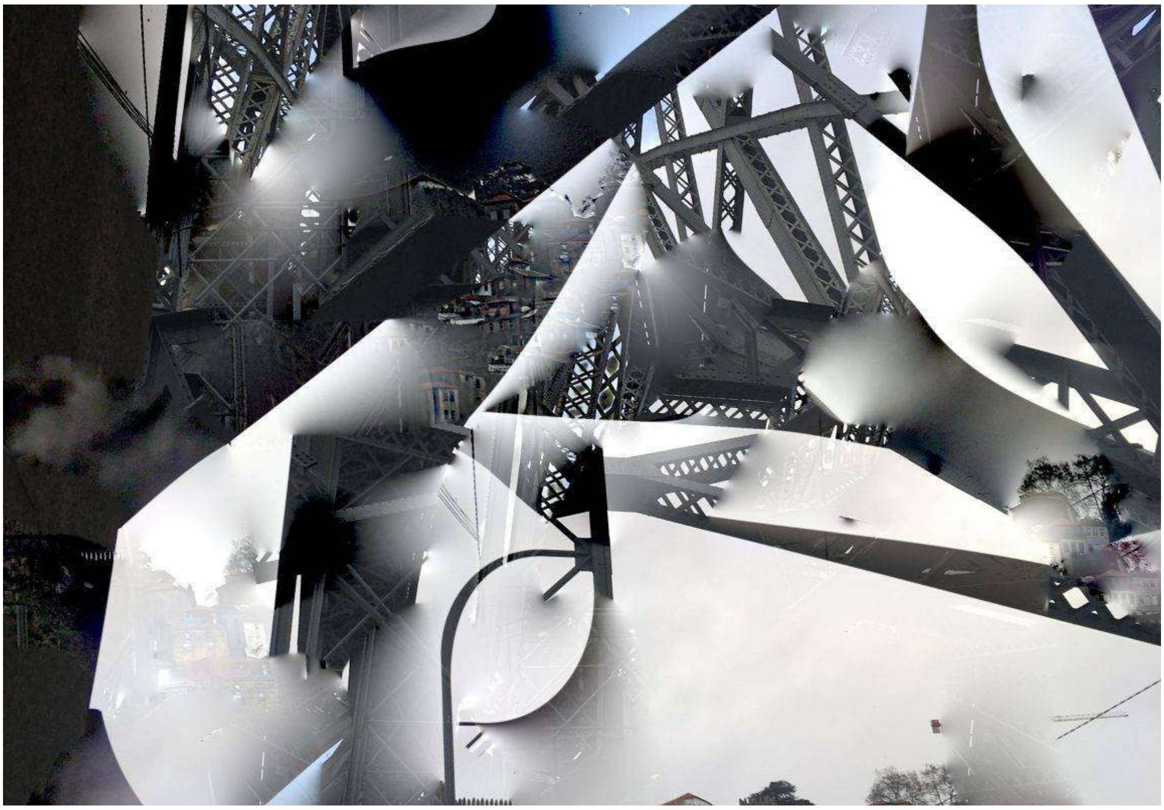
☰



ALAN N. SHAPIRO 2024-08-02

# HAYLES ON WRITING AND SOFTWARE CODE

GENERICSCIENCE, MASHINES CODE, INFORMATION, SIMULATION, WRITING

In his book *Does Writing Have a Future?*, the luminary media theorist Vilém Flusser lays out an intellectual project of connecting the future of software code with the history of writing. The code of the future will become more like the writing of the past – or rather, in the future there will be an as-yet-concealed hybrid of code and writing. My contention is that the so-called discrete logic of identities and differences or "computable numbers" that enabled the invention of the digital-binary computer – e.g., all software state machine operations ultimately change voltage, manipulate bits, and are stepwise; all variable names and variable values are different from each other; each instruction or line of code has a definite unambiguous meaning – was based on the linguistic-philosophical idea of a purely "formal language" that suppresses the poetic, musical, ambivalent, and resonant – and semantic, syntactic, and semiotic – qualities of human languages.

As the history of programming languages proceeds further, and in the spirit of Creative Coding, one can put forth the hypothesis that the dimension of human language will be reappearing more and more in programming languages.

Hayles' work and vision point profoundly in that direction. She has written incisively and presciently on the connection between literature and technology, on digital literature as literary practice, on experimental novels, on digital poetry, and on the relationship of print books to software code.620 She writes about hypertext stories, interactive fiction, metaphoric networks, the relation between narrative and database, the intermediation between page and screen, and the ways that electronic literature transforms computational practice. "Writing machines" are techno-texts where a work self-reflexively interrogates the media of inscription which produced it, in a recursive act of bringing its imaginative world to life in dialogue with "the material apparatus embodying that creation as a physical presence."

In *My Mother Was a Computer: Digital Subjects and Literary Texts*, Hayles compares the "worldviews" of speech, writing, and code. The interaction or fusion of human language and code occurs in the world more and more "in millions of encounters every day." Human experience and the subjective sense of self are constituted through bits as well as through words. In explicating the speech and writing *Weltanschauungen*, Hayles takes as her references the theories of semiotics of Saussure and of "grammatology" of Jacques Derrida (which was based on a critique of Saussure), respectively. Derrida focuses on the negative "linguistic concept of difference without positive terms," taking apart Saussure's dualistic metaphysics of signifier and signified. Value undergoes the negative critique of the subversive and differential play of language. In Derrida's "différance," meaning is always differing and deferred. Can there be a "deconstructionist" writing of software code? Low-level programming languages such as machine language, assembler, and procedural-functional languages are "close to the machine" and operate in combinatorial ways on voltages and bits. Yet Hayles departs from Kittler's position in his essay "There is No Software" that all levels of the code translate or "boil down" to what happens at the hardware level. She observes:

As the system builds up levels of programming languages such as compilers, interpreters, scripting languages, and so forth, they develop functionalities that permit increasingly greater ambiguities in the choices permitted or tolerated… Only at the high level of object-oriented languages such as C++ does code recuperate the advantages of citability and iterability (i.e., inheritance and polymorphism, in the discourse of programming language) and in this sense become 'grammatological'.

The software layer is the translation between human and machine language. This traversal actuality of translation and the corporeality of the human factor make code to a certain degree – and potentially more so in the future – sovereign from the hardware-level bit-manipulation. OO programming languages – such as C++, C#, and Java – are consciously imitative of human language. They are languages of modeling, simulation, and virtuality that replicate real-world processes and environments (from banking applications to "virtual worlds") in software. The OO design, which identifies the nouns or "classes" or "objects" in the given problem domain, and the verbs or process relations between the objects, is, in a sense, already the code. “The problem and the solution are both expressed in equivalent terms.”

In the final part of this book, I will discuss the Creative Coding movement (software development tools and projects for artists, designers, and creative people) and its implications for the future of computer languages. The object-oriented paradigm is also part of the practice of writing software code becoming an expressive media. Hayles writes:

As computers are increasingly understood (and modeled after) “expressive mediums” like writing, they begin to acquire the familiar and potent capability of writing not merely to express thought but actively to constitute it. As high-level computer languages move closer to natural languages, the processes of intermediation by which each affects the other accelerate and intensify.

OO programming languages need to get closer to human languages. At present, they assume the existence of a “real world” and so-called “real world” processes. Software development is the practice of modeling these offline processes in a virtual space. This alleged “real world” is in fact the realm of simulacra and simulation. Modeling what one believes to be “real-world” processes which are indeed simulations amounts to practicing the simulation of a simulation.

In the “codework” software poetry of Alan Sondheim, and in generative art live “code performances,” code and language interpenetrate, and the written program as code and poetry executes. In the later history of the computer, we are moving beyond the founding binary logic of identity and difference to a fecund enmeshment in the complexified and embodied tension between those two terms. The binary code of intelligent machines becoming more like the resonant language of intelligent humans is indeed the writing of SF in the “new real.”

taken from here: http://www.alan-shapiro.com/hayles-on-writing-and-software-code/

← PREVIOUS NEXT →

---

META

CONTACT
FORCE-INC/MILLE PLATEAUX

IMPRESSUM
DATENSCHUTZERKLÄRUNG

TAXONOMY

CATEGORIES
TAGS
AUTHORS
ALL INPUT

SOCIAL

FACEBOOK
INSTAGRAM

TWITTER